[redacted]

UIUCDCS-R-79-985

UILU-ENG 79 1734

On The Complexity of Inferring Join Dependencies

by

David Maier

Yehoshua Sagiv

August 1979

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-79-985

On the Complexity of Inferring Join Dependencies

David Maier
Department of Computer Science
State University of New York
at Stony Brook
Stony Brook, New York 11794

Yehoshua Sagiv+
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

August 1979

(+) The work of this author was supported in part by the National Science Foundation under grant MCS-77-22830

ABSTRACT

It is shown that deciding whether a set of functional dependencies and one join dependency implies another join dependency is NP-complete. It is also shown that deciding whether a JD-rule can be applied to a tableau T is NP-complete. This problem is NP-complete even if T can be obtained from a tableau corresponding to a join dependency by applying some FD-rules. As a result, it follows that computing the join of several relations is NP-hard.

CR categories: 4.33, 5.25

Key words and phrases: functional dependency, multivalued dependency, join dependency, join, membership algorithm, NP-complete, relational database.

1. Introduction

The relational model for databases [Cod] uses dependencies as a semantic tool for expressing constraints that the data must satisfy. Functional dependencies and join dependencies (that include multivalued dependencies as a special case) are examples of such dependencies. A utilization of these dependencies in the design of relational databases depends upon the ability to develop membership algorithms, that is, algorithms for deciding whether a set of dependencies Σ implies another dependency σ. Several efficient membership algorithms are known if all the dependencies are functional or multivalued [Bel,BeB,Gal,HIT,Sag], and an exponential time and space algorithm exists for functional and join dependencies [MMS].

In this paper we show that if σ is a join dependency, and Σ is a set of functional dependencies and one join dependency, then deciding whether Σ implies σ is NP-complete. As a by-product of this result, we show that the problem of deciding whether a JD-rule can be applied to a tableau T, and the problem of deciding whether a relation r does not obey a join dependency are NP-complete. The first problem is NP-complete even if T can be obtained from a tableau corresponding to a join dependency by applying some FD-rules. Another by-product is a proof that deciding whether a relation r is not the join of relations $r_1,\ldots,r_n$ is NP-hard. It is easy to give examples in which the join of $r_1,\ldots,r_n$ has an exponential size (measured as a function of the space needed to write down $r_1,\ldots,r_n$). Therefore, this result indicates that an algorithm for computing the join of $r_1,\ldots,r_n$ whose running time is

polynomial in the size of the output (i.e., the space needed to write down the join of $r_1,\ldots,r_n$) is unlikely to exist. A similar result is given in [HLY]. However, our result is stronger, since we assume that $r_1,\ldots,r_n$ are projections of some universal instance I.

A recent result [Yan] shows that if σ is a functional or a multivalued dependency, then deciding whether a set Σ of functional and join dependencies implies σ can be done in polynomial time. Thus, the only remaining open problem is to find a lower bound on the complexity of deciding whether a set of join dependencies implies another join dependency. It is interesting to note that the only known algorithm for the more restricted problem of deciding whether a set of multivalued dependencies implies a join dependency is exponential in time and space, and there is no known lower bound [ABU].

2. Basic Definitions

A _relation_ is a two-dimensional table in which columns correspond to _attributes_, and rows correspond to _records_ or _tuples_. Each attribute has an associated _domain_ of values, and a tuple is viewed as a mapping from the attributes to their domains. If r is a relation, μ is a tuple of r, and X is a set of attributes, then $\mu[X]$ denotes the values of μ in the X-columns. A set of attributes labeling the columns of a relation is called a _relation scheme_. If R is a set of attributes labeling the columns of a relation r, then r is said to be _defined on_ R. We use the letters A,B,C,... to denote attributes, and the letters

...,R,S,...,X,Y,Z to denote sets of attributes (i.e., relation schemes). A set of attributes is written as a string attributes (e.g., ABCD is the set {A,B,C,D}), and the union of sets of attributes X and Y is written XY. In this paper we assume that all the attributes are drawn from a universal set of attributes U.

A _functional dependency_ (abbr. FD) [Arm,Cod] is a statement of the form $X \rightarrow Y$, where both X and Y are sets of attributes. The FD $X \rightarrow Y$ _holds_ in a relation r, if for all tuples μ and ν of r, if $\mu[X] = \nu[X]$, then $\mu[Y] = \nu[Y]$.

Let $R_1,...,R_q$ be relation schemes, and let r be a relation on $\bigcup_{i=1}^{q} R_i$. Suppose that $\mu_1,...,\mu_q$ are q tuples of r (not necessarily distinct). The tuples $\mu_1,...,\mu_q$ are _joinable_ on $R_1,...,R_q$ with a _result_ ν, if there exists a mapping ν defined on $\bigcup_{i=1}^{q} R_i$ such that for all $1 \le i \le q$, $\mu_i[R_i] = \nu[R_i]$. A _join dependency_ (abbr. JD) [Ris] is a statement of the form $*[R_1,...,R_q]$, where each R_i is a relation scheme. The JD $*[R_1,...,R_q]$ _holds_ in a relation r defined on $\bigcup_{i=1}^{q} R_i$ if whenever tuples $\mu_1,...,\mu_q$ of r are joinable on $R_1,...,R_q$ with a result ν, then ν is also a tuple of r. The JD $*[R_1,...,R_q]$ is _defined on_ the relation scheme $\bigcup_{i=1}^{q} R_i$.

A _multivalued dependency_ (abbr. MVD) [BFH,Fag,Zan] is a JD with at most two relation schemes. An MVD $*[R_1,R_2]$ is also written $R_1 \cap R_2 \twoheadrightarrow R_1$ (or equivalently $R_1 \cap R_2 \twoheadrightarrow R_2$). Conversely, the MVD $X \twoheadrightarrow Y$ defined on U can be written as the JD $*[XY,XZ]$, where $Z = U - X - Y$. Both FD's and MVD's have a complete set of inference

rules [Arm,BFH], and polynomial time membership algorithms [Bel,BeB,Gal,HIT,Sag]. An MVD $X \twoheadrightarrow Y$ holds in a relation r if and only if $X \twoheadrightarrow Y - X$ holds in r [Fag]. Therefore, we can assume that in an MVD $X \twoheadrightarrow Y$ the left and right sides (i.e., X and Y) are disjoint.

Let $r_1,\dots,r_n$ be relations defined on $R_1,\dots,R_n$, respectively. The <u>join</u> of $r_1,\dots,r_n$, written $\mathop{*}\limits_{i=1}^{n} r_i$, is

$$\{\mu \mid \text{there are tuple } \mu_i \varepsilon r_i \ (1 \le i \le n) \text{ such that } \mu_1,\dots,\mu_n \text{ are joinable on } R_1,\dots,R_n \text{ with a result } \mu\}$$

Let Σ be a set of JD's, and let σ be a JD or an FD. We assume that all the JD's are defined on U. The dependency σ is a <u>consequence</u> of Σ (or σ is <u>implied</u> by Σ) if and only if for all relations r on U, the dependency σ holds in r if all the dependencies of Σ hold in r.

Let Σ be a set of dependencies. A convenient way of representing all the MVD's with a fixed left side that are implied by Σ is by constructing the dependency. The <u>dependency</u> <u>basis</u> of a set of attributes X is a partition of U into pairwise disjoints subsets of attributes $X,Y_1,\dots,Y_n$ such that

(1) $X \twoheadrightarrow Y_i$ is implied by Σ $(1 \le i \le n)$, and

(2) if $X \twoheadrightarrow Y$ is implied by Σ, then Y is a union of some of the Y_i's.

The existence of the dependency basis follows from the inference rules for MVD's [Fag]. If Σ contains only FD's and MVD's, then the dependency basis can be constructed in polynomial time [Bel,Gal,HIT,Sag].

A <u>tableau</u> [ABU,ASU] is a two-dimensional matrix in which columns correspond to attributes. The rows of a tableau consist of variables of

the following types

(1) distinguished variables, usually denoted by subscripted a′s, and

(2) nondistinguished variables, usually denoted by subscripted b′s.

A variable cannot appear in more than one column, and in each column there is exactly one distingushed variable.

A JD $*[R_1,\ldots,R_q]$ has a corresponding tableau T as follows. For each R_i, tableau T has a row w_i with distinguished variables exactly in the R_i-columns, and distinct nondistinguished variable in the rest of the columns. We can also view a tableau as a relation over the domain of distinguished and nondistinguished variables. Note that rows $w_1,\ldots,w_q$ of T are joinable on $R_1,\ldots,R_q$, and the resulting row consists only of distinguished variables.

Example 1: Consider the JD *[AB,BCD,AD]. The tableau T_1 corresponding to this JD is

$$\begin{array}{cccc} A & B & C & D \\ \end{array}$$
$$\begin{vmatrix} a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & a_3 & a_4 \\ a_1 & b_4 & b_5 & a_4 \end{vmatrix}$$

[]

Let Σ be a set of FD′s and JD′s. Each dependency in Σ has an associated rule that can be applied to any tableau T as follows.

(1) FD-Rules. An FD $X \to Y$ in Σ has an associated rule for equating variables of T as follows. Suppose that rows w_1 and w_2 of T agree in all the X-columns, but disagree in an A-column, where A is an attribute of Y. If one of w_1 and w_2 has a distinguished variable in its A-column,

then rename the two rows so that w_1 is that row. The FD-rule for $X \to Y$ replaces all occurrences of the variable appearing in the A-column of w_2 with the variable appearing in the A-column of w_1.

(2) JD-Rules. A JD $*[S_1,\ldots,S_p]$ in Σ has an associated rule for adding rows to T as follows. If rows $w_1,\ldots,w_p$ of T are joinable on $S_1,\ldots,S_p$ with a result w, and w is not already in T, then w is added to T.

Each one of The above rules transforms a tableau T to another tableau T′. The rules can be applied repeatedly to a tableau T only a finite nuber of times, and the result is unique (up to renaming of non-distinguished variables) [MMS]. The chase of T under Σ, denoted $\mathrm{chase}_\Sigma(T)$, is the tableau obtained by applying the rules associated with Σ to T until no rule can be applied anymore. Let σ be a JD with a corresponding tableau T_σ. The JD σ is a consequence of Σ if and only if $\mathrm{chase}_\Sigma(T_\sigma)$ contains a row consisting only of distinguished variables [MMS].

Example 2: Let $\Sigma = \{*[AB,BCD,ABD], A \to B, C \to A\}$, and let σ be the JD $*[AB,BCD,AD]$ whose corresponding tableau is given in Example 1. The FD-rule for $A \to B$ can be applied to the first and third rows of the tableau in Example 1, and hence b_4 is identified with a_2. The resulting tableau is

$$\begin{array}{cccc} A & B & C & D \\ \end{array}$$
$$\begin{vmatrix} a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & a_3 & a_4 \\ a_1 & a_3 & b_5 & a_4 \end{vmatrix}$$

The first, second, and third rows of the above tableau are joinable on AB,BCD,ABD with a result (a_1,a_2,a_3,a_4). Thus, applying the JD-rule for *[AB,BCD,ABD] produces the tableau

$$\begin{array}{cccc} A & B & C & D \\ \end{array}$$
$$\begin{vmatrix} a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & a_3 & a_4 \\ a_1 & a_2 & b_5 & a_4 \\ a_1 & a_2 & a_3 & a_4 \end{vmatrix}$$

Applying the FD-rule for $C \rightarrow A$ to the second and fourth rows of the above tableau identifies b_3 with a_1. As a result the second row becomes identical to the fourth row, and hence it can be omitted. The resulting tableau is

$$\begin{array}{cccc} A & B & C & D \\ \end{array}$$
$$\begin{vmatrix} a_1 & a_2 & b_1 & b_2 \\ a_1 & a_2 & b_5 & a_4 \\ a_1 & a_2 & a_3 & a_4 \end{vmatrix}$$

No rule for Σ can be applied to the above tableau []

3. NP-Completeness Results Concerning Join Dependencies

3.1 Boolean Expressions and Tableaux

All the results use almost the same reduction from the 3-satisfiability (3-SAT) problem, shown NP-complete in [C]; see also [K,GJ]. Let $Q = F_1 \cdots F_m$ be a Boolean expression in conjunctive normal form, where the F_j's are clauses of three literals each, and $x_1, x_2, \ldots, x_n$ are all the variables appearing in this expression. We denote the variables appearing in a clause F_j by x_{j_1}, x_{j_2}, and x_{j_3}. We assume that $n \geq 4$ (and hence $m > 1$), and each variable appears in at least two clauses. Note that if $n < 3$, then the satisfiability of Q can be decided in linear time; and if a variable appears in only one clause, then this clause is always satisfied and, hence, it can be omitted. Thus, this variant of the 3-SAT problem is NP-complete.

We now show how Q is used to construct two tableaux that correspond to join dependencies. These tableaux are similar to those used in the NP-completeness proofs given in [ASU]. Each one of them has $(m+3n+2)$ columns. The first m columns correspond to the clauses $F_1, \ldots, F_m$, and they are labeled by the attributes $E_1, \ldots, E_m$. The next 3n columns are divided into three blocks of n columns each. The n columns in each block correspond to the variables $x_1, \ldots, x_n$. The columns of the three blocks are labeled by A_i's, B_i's, and C_i's, respectively. The last two columns are labeled by D_1 and D_2. The first tableau, denoted by S, represents the m clauses. For each clause F_j containing the variables x_{j_1}, x_{j_2}, and x_{j_3}, tableau S has a row s_j as follows. Row s_j has distinguished variables in the columns for E_j, A_{j_1}, A_{j_2}, A_{j_3}, and D_1. All

the other columns have distinct nondistnigushed variables. The tableau S has one more row, denoted by s_{m+1}, with distinguished variables in all the E, B, C, and D_2 columns (the rest of the columns have distinct nondistinguished variables). Let S_j be the relation scheme corresponding to row s_j of S ($1 \leq j \leq m+1$). That is, S_j contains all the attributes labeling columns in which s_j has distinguished variables. Thus, the tableau S corresponds to the JD $*[S_j, \ldots, S_{m+1}]$.

The second tableau, denoted by T, represents truth assignments under which clauses of Q are true. For every F_j ($1 \leq j \leq m$), tableau T has seven rows that represent all the truth assignments under which F_j is true. If ζ is a truth assignment under which F_j is true, then T contains a row w as follows. For $1 \leq i \leq 3$, if x_{j_i} is assigned 1 under ζ, row w has a distinguished variable in the B_{j_i}-column; otherwise, w has a distinguished variable in the C_{j_i}-column. Row w has distinguished variables also in the E_j-column and in the D_1-column. The tableau T has two additional rows, denoted by u and v. Row u has distinguished variables in all the E, B, C, and D_1 columns (excatly as row s_{m+1} of S). Row v has distinguished variables in all the A and D columns. All the other columns of rows of T contain distinct nondistinguished variables.

<u>Example 3</u>: Consider the Boolean expression

$$(x_1 + \overline{x}_2 + x_3)(x_1 + \overline{x}_3 + x_4)(\overline{x}_1 + x_2 + \overline{x}_4).$$

By a slight abuse of notation, we denote the distinguished variable in each column by an a (without a subscript). The dots stand for distinct nondistinguished variables. The tableau S is

	E_1	E_2	E_3	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4	C_1	C_2	C_3	C_4	D_1	D_2
s_1	a	.	.	a	a	a	.	.	.	.	.	.	.	.	.	a	.
s_2	.	a	.	a	.	a	a	.	.	.	.	.	.	.	.	a	.
s_3	.	.	a	a	a	.	a	.	.	.	.	.	.	.	.	a	.
s_4	a	a	a	.	.	.	.	a	a	a	a	a	a	a	a	.	a

The tableau T is given in Figure 1. []

Let Σ be a set of dependencies that consists of the JD $*[S_1,\ldots,S_{m+1}]$, and the FD's $B_iD_1 \rightarrow A_i$, $C_iD_1 \rightarrow A_i$, and $D_1D_2 \rightarrow A_i$ (for $1 \leq i \leq n$); and let σ be the JD corresponding to the tableau T.

We will show that σ is a consequence of Σ if and only if Q is satisfiable. The proof is an analysis of the computation of $\text{chase}_\Sigma(T)$. Since the rules associated with Σ can be applied to T in any order, we start by applying the FD-rules. The FD-rules for $D_1D_2 \rightarrow A_i$ $(1 \leq i \leq n)$ cannot be applied, since no two rows of T agree in the columns for D_1 and D_2. The application of the other FD-rules modifies only the A-columns of T. Note that rows u and v of T are not affected by this modification. After all possible applications of FD-rules to T, each A_i-column is going to have exactly two repeated(1) nondistinguished variables, say b_i and $\bar{b}_i$ $(1 \leq i \leq n)$. The variable b_i results from the application of the FD-rules for $B_iD_1 \rightarrow A_i$, and can be viewed as representing the truth value 1. The variable $\bar{b}_i$ results from the application of the FD-rules for $C_iD_1 \rightarrow A_i$, and can be viewed as representing the truth value 0. A row w of T representing a truth assignment for a clause F_j (with variables x_{j_1}, x_{j_2}, and x_{j_3}) is going to have b_{j_i} in the A_{j_i}-column, if x_{j_i} is true; otherwise, it is going to have $\bar{b}_{j_i}$ in this column $(1 \leq i \leq n)$.

---
(1) A variable is repeated if it appears in more than one row.

E_1	E_2	E_3	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4	C_1	C_2	C_3	C_4	D_1	D_2
a	.	.	.	.	.	.	.	.	.	.	a	a	a	.	a	.
a	.	.	.	.	.	.	.	.	a	.	a	a	.	.	a	.
a	.	.	.	.	.	.	.	a	a	.	a	.	.	.	a	.
a	.	.	.	.	.	.	a	.	.	.	.	a	a	.	a	.
a	.	.	.	.	.	.	a	.	a	.	.	a	.	.	a	.
a	.	.	.	.	.	.	a	a	.	.	.	.	a	.	a	.
a	.	.	.	.	.	.	a	a	a	.	.	.	.	.	a	.
.	a	.	.	.	.	.	.	.	.	.	a	.	a	a	a	.
.	a	.	.	.	.	.	.	.	.	a	a	.	a	.	a	.
.	a	.	.	.	.	.	.	.	a	a	a	.	.	.	a	.
.	a	.	.	.	.	.	a	.	.	.	.	.	a	a	a	.
.	a	.	.	.	.	.	a	.	.	a	.	.	a	.	a	.
.	a	.	.	.	.	.	a	.	a	.	.	.	.	a	a	.
.	a	.	.	.	.	.	a	.	a	a	.	.	.	.	a	.
.	.	a	.	.	.	.	.	.	.	.	a	.	a	a	a	.
.	.	a	.	.	.	.	.	.	.	a	a	.	a	.	a	.
.	.	a	.	.	.	.	.	.	a	.	a	.	.	a	a	.
.	.	a	.	.	.	.	.	.	a	a	a	.	.	.	a	.
.	.	a	.	.	.	.	a	.	.	.	.	.	a	a	a	.
.	.	a	.	.	.	.	a	.	a	.	.	.	.	a	a	.
.	.	a	.	.	.	.	a	.	a	a	.	.	.	.	a	.
a	a	a	.	.	.	.	a	a	a	a	a	a	a	a	.	a
.	.	.	a	a	a	a	.	.	.	.	.	.	.	.	a	a

Figure 1

Thus, the truth assignment represented by w is now given in the A-columns of w. It is easy to show that no further applications of FD-rules are possible. Let T′ be the tableau obtained by applying the FD-rules to T.

Lemma 1: Suppose that rows $w_1, \ldots, w_{m+1}$ of T′ are joinable on $S_1, \ldots, S_{m+1}$ with a result w, and w is not in T. Then w_{m+1} is u, and for all $1 \le j \le m$, row w_j is a row of T representing a truth assignment for F_j.

Proof: If all the w_i′s are identical, then w is the same row as the w_i′s and, hence, it is in T′. Therefore, it suffices to show that if

either w_{m+1} is not u or some w_j $(j \neq m+1)$ is not a row representing a truth assignment for F_j, then all the w_i's are identical.

<u>Claim 1</u>: If row w_{m+1} or row w_i (for some $1<i<m$) has in the E_i-column a nondistinguished variable that appears nowhere else in T', then w_i and w_{m+1} are identical.

Claim 1 follows from the fact that for all $1<i<m$, rows w_i and w_{m+1} agree in the E_i-column, because both S_i and S_{m+1} contain E_i.

<u>Claim 2</u>: If some w_j $(j \neq m+1)$ is u, then for all $1<i<m$, row w_i is u.

Claim 2 follows from the fact that for all $1<i<m$, the relation scheme S_i contains the attribute D_1, and u has in the D_1-column a nondistinguished variable appearing nowhere else in T'.

Suppose that w_{m+1} is v. But v has in each E_i-column a distinct nondistinguished variables appearing nowhere else in T', and so by Claim 1, every w_i is v. So suppose that w_{m+1} is a row representing a truth assignment for some F_k. Therefore, row w_{m+1} has a distinguished variable in the E_k-column, and in all the other E-columns it has distinct nondistinguished variables appearing nowhere else in T'. By Claim 1, for all $i \neq k$, row w_i and w_{m+1} are identical. Row w_k must have a distinguished variable in the E_k-column, since w_{m+1} has a distinguished variable in this column and both S_k and S_{m+1} contain E_k. By Claim 2, row w_k cannot be u, because there is a row w_j $(j \neq m+1)$ that is not u (since $m>1$). Thus, all the w_i's $(i \neq k)$ are equal to a row of T' representing a truth assignment for F_k, and w_k is also a row represent-

ing a truth assignment for F_k. But every variable x_q appears in more than one clause and, hence, the pattern of the distinguished variables in the A-columns of tableau S implies that w_k represents the same truth assignment as all the other w_i's. That is, all the w_i's are identical.

So far we have shown that if w_{m+1} is not u, then all the w_i's are identical. Now suppose that some w_j is not a row representing a truth assignment for F_j. If w_j is u, then Claim 2 implies that for all $1<i<m$, row w_i is u. But w_{m+1} is also u, and so all the w_i's are identical. If w_j is either v or a row representing a truth assignment for some F_k $(j \neq k)$, then w_j has in the E_j-column a nondistinguished variable appearing nowhere else in T′, and so by Claim 1, rows w_j and w_{m+1} are identical. But w_{m+1} is u, and so Claim 2 implies that all the w_i's are identical. []

<u>Corollary 2</u>: Rows $w_1,\ldots,w_{m+1}$ of T′ are joinable on $S_1,\ldots,S_{m+1}$ with a result w not in T′ if and only if Q is satisfiable.

<u>Proof</u>: <u>Only if</u>. By Lemma 1, row w_j $(1<j<m)$ represents the following truth assignment for F_j. If x_{j_i} is a variable of F_j, and the A_{j_i}-column of w_j has the repeated nondistinguished variable b_{j_i}, then x_{j_i} is assigned 1. If the A_{j_i}-column of w_j has the repeated nondistinguished variable $\bar{b}_{j_i}$, then x_{j_i} is assigned 0. Under this truth assignment F_j is true. But the pattern of the distinguished variables in the A-columns of tableau S implies that in this case there is a truth assignment ψ for all the variables $x_1,\ldots,x_n$ such that for all $1<j<m$, the truth assignment ψ agrees with the truth assignment represented by

w_j on the variables of F_j. Hence, each F_j is true under ψ, and Q is satisfiable.

_If_. Suppose that ψ is a truth assignment that satisfies Q. For all $1 \leq j \leq m$, let w_j be the row of T′ representing the truth assignment for F_j that agrees with ψ on the variables of F_j; and let w_{m+1} be row u of T′. It is easy to show that the rows $w_1, \ldots, w_{m+1}$ are joinable on $S_1, \ldots, S_{m+1}$ with a result w not in T′. []

Lemma 3: The JD σ (corresponding to T) is a consequence of Σ if and only if Q is satisfiable.

Proof: _Only if_. By Corollary 2, if Q is not satisfiable, then the only JD-rule for Σ cannot be applied to T′. Therefore, $\text{chase}_\Sigma(T)$ is the result of applying the FD-rules to T, i.e., $\text{chase}_\Sigma(T) = T'$. This chase does not contain a row with only distinguished variables and, hence, σ is not a consequence of Σ.

_If_. Suppose that Q is satisfiable. By Lemma 1 and Corollary 2, an application of the JD-rule for Σ to T′ adds a row w that has distinguished variables in all the E, B, C, and D columns. We can apply the FD-rules for $D_1D_2 \to A_j$ $(1 \leq j \leq n)$ to w and v (the last row of T′), and the result is a row with only distinguished variables. Thus, σ is a consequence of Σ. []

3.2 NP-Completeness Results Concerning Applications of JD-Rules and Testing Whether Relations Obey Join Dependencies.

Theorem 4: The problem of deciding whether a JD-rule can be applied to a tableau U is NP-complete. This problem is NP-complete even if U can be obtained from a tableau corresponding to a JD by applying some FD-rules.

Proof: At first we will show that the problem is in NP. Suppose we have to decide whether the JD-rule for a JD $*[R_1,\ldots,R_q]$ can be applied to a tableau U. We nondeterministically choose q rows $w_1,\ldots,w_q$ of U, and check in polynomial time whether they are joinable on $R_1,\ldots,R_q$ with a result w not in U.

To show that the problem is complete in NP, the 3-SAT problem can be reduced to this problem as described in Section 3.1. That is, given an instance Q of the 3-SAT problem, we construct the JD $*[S_1,\ldots,S_m]$ and the tableau T. By applying some FD-rules to T, we obtain the tableau T′. By Corollary 2, the JD-rule for $*[S_1,\ldots,S_m]$ can be applied to T′ if and only if Q is satisfiable. []

Corollary 5: It is NP-complete whether a JD $*[R_1,\ldots,R_q]$ does not hold in a relation r.

Proof: The problem is in NP, since we can nondeterministically find q tuples of r that are joinable on $R_1,\ldots,R_q$ with a result that is not a tuple of r.

To show that the problem is complete in NP, we can view the tableau T' as a relation r (by replacing each variable with a distinct constant). By Corollary 2, the JD $*[S_1,\ldots,S_m]$ does not hold in r if and only if Q is satisfiable. []

3.3 An NP-Completeness Result for Inferring Join Dependencies

Theorem 6: Let Γ be a set of FD's and one JD, and let γ be another JD. The problem of deciding whether γ is a consequence of Γ is NP-complete.

Proof: Let $*[R_1,\ldots,R_q]$ be the only JD in Γ. At first we show that the problem is in NP. Let U be a tableau and suppose that $\text{chase}_\Gamma(U)$ can be obtained from U by using only the JD-rule for Γ. The following claim shows that any row of $\text{chase}_\Gamma(U)$ (that is not in U) can be obtained by a single application of the JD-rule for Γ to some rows of U.

Claim 1: If a tableau U' is obtained by repeatedly applying the JD-rule for Γ to a tableau U, then any row of U' is the result of joining some rows of U on $R_1,\ldots,R_q$.

In order to prove this claim, suppose that the JD-rule for Γ is applied only to the original rows of U until it cannot be applied anymore. Let the resulting tableau be $\overline{U}$. It suffices to show that the JD-rule for Γ cannot be applied to $\overline{U}$. So suppose that the JD-rule can be applied to $\overline{U}$. That is, there are rows $w_1,\ldots,w_q$ of $\overline{U}$ that are joinable on $R_1,\ldots,R_q$ with a result w not in $\overline{U}$. If some w_i is not in U,

then there are rows $v_1,\ldots,v_q$ in U that are joinable on $R_1,\ldots,R_q$ with a result w_i. But w_i and v_i agree on R_i and, hence, w_i can be replaced with v_i. That is, $w_1,\ldots,w_{i-1},v_i,w_{i+1},\ldots,w_q$ are joinable on $R_1,\ldots,R_q$ with a result w. It follows that every w_i that is not in U can be replaced with some row in U, and the resulting rows are joinable on $R_1,\ldots,R_q$ with a result w. Therefore, w is also in $\overline{U}$.

Now suppose that no FD-rule for Γ can be applied to a tableau U, but some FD-rules for Γ can be applied to a tableau U′, where U′ is obtained from U by applying the JD-rule for Γ several times. That is, there are rows v and w of U′ such that some FD-rule for Γ can be applied to v and w. By Claim 1, rows v and w can be generated by applying the JD-rule to rows of U (unless they are already in U). By using a nondeterministic algorithm, rows v and w can be obtained in polynomial time in no more than two applications of the JD-rule for Γ. Therefore, in order to generate any row of $chase_\Gamma(U)$, we can always find a sequence of applications of the rules for Γ in which the JD-rule is never used more than twice in a row. Let n be the number of distinct variables in U. Each application of an FD-rule reduces the number of distinct variables by one. Thus, the FD-rules can be applied to U no more than n times. Since each application of an FD-rule is preceded by no more than two applications of the JD-rule for Γ, we can generate any row of $chase_\Gamma(U)$ in O(n) applications of the rules for Γ. In particualr, we can use a nondeterministic algorithm to generate the row consisting only of distinguished variables (if this row is indeed in $chase_\Gamma(U)$) in O(n) applications of the rules for Γ.

The following is a nondeterministic polynomial time algorithm that returns "Yes" if γ is a consequence of Γ. The tableau for γ is denoted by V.

(1) Nondeterministically create two rows v_1 and v_2 such that each v_i is either a row of V or can be obtained by joining some rows of V on $R_1, \ldots, R_q$.

(2) If either v_1 or v_2 consists only of distinguished variables, then return "Yes".

(3) Add v_1 and v_2 to V (if they are not already there).

(4) Apply the FD-rules to V until no FD-rule can be applied. If at least one FD-rule has been applied, then go to (1).

Steps (1)-(3) require nondeterministic linear time. Step (4) requires (deterministic) polynomial time [ABU]. Each application of an FD-rule reduces the number of distinct variables in V by one, and Step (1) is repeated only after an application of some FD-rule. Therefore, no more than O(n) rows are added to V, and the algorithm has a nondetem-inistic polynomial running time.

It remains to be shown that the problem is NP-complete. The 3-SAT problem can be reduced to this problem as described in Section 3.1, and the NP-completeness follows from Lemma 3. []

3.4 An NP-Hard Result for Computing the Join of Several Relations

In this section we show that computing the join of several relations is a hard problem (even if the relations come from a universal instance). We assume familiarity with the definition of the join operator, and the correspondence between tableaux and relational expressions (cf. [ASU]). It should be noted that a similar result is stated in [HLY]. However, our result is stronger, since we assume that the relations are obtained by projection from a universal instance.

Theorem 7: Let E be a relational expression with the join as the only operator, let I be a universal instance, and let r be a relation. The problem whether E(I) ≠ r is NP-hard. (E(I) is the value of the expression E for the instance I.)

Proof: We can view the tableau T′ of Section 3.1 as a universal instance, and the tableau S as representing the relational expression $\mathop{*}_{i=1}^{m} S_i$. Thus the 3-SAT problem can be reduced to this problem in the following way. Given a Boolean expression Q, we construct the relational expression $\mathop{*}_{i=1}^{m} S_i$ corresponding to the tableau S of Section 3.1. The instance I is obtained from the tableau T′ by replacing each variable of T′ with a distinct element from the domain of the corresponding attribute. The relation r is the same as the instance I. By Corollary 2, the relation r is not the value of $\mathop{*}_{i=1}^{m} S_i$ for I if and only if Q is satisfiable. []

References

[ABU] Aho, A. V., C. Beeri, and J. D. Ullman, "The Theory of Joins in Relational Databases," ACM Trans. on Database Systems, Vol. 4, No. 3 (Sept., 1979), pp. 297-314.

[ASU] Aho, A. V., Y. Sagiv, and J. D. Ullman, "Equivalences Among Relational Expressions," SIAM J. Computing, Vol. 8, No. 2 (May, 1979), pp. 218-246.

[Arm] Armstrong, W. W., "Dependency Structures of Database Relationships," Proc. IFIP 74, North Holland, 1974, pp. 580-583.

[Bel] Beeri, C., "On the Membership Problem for Multivalued Dependencies in Relational Databases," to appear in ACM Trans. on Database Systems.

[BeB] Beeri, C., and P. A. Bernstein, "Computational Problems Related to the Design of Normal Form Relational Schemas," ACM Trans. on Database Systems, Vol. 4, No. 1 (March, 1979), pp. 30-59.

[BFH] Beeri, C., R. Fagin, and J. H. Howard, "A Complete Axiomatization for Functional and Multivalued Dependencies in Database Relations," Proc. ACM-SIGMOD Int. Conf. on Management of Data, Toronto, Aug., 1977, pp. 47-61.

[Cod] Codd, E. F., "A Relational Model for Large Shared Data Banks," Comm. ACM, Vol. 13, No. 6 (June, 1970), pp. 377-387.

[Coo] Cook, S. A., "The Complexity of Theorem Proving Procedures," Proc. 3rd Annual ACM Symp. on Theory of Computing, May, 1971, pp. 151-158.

[Fag] gin, R., "Multivalued Dependencies and a New Normal Form for Relational Databases," ACM Trans. on Database Systems, Vol. 2, No.

3 (Sept., 1977), pp. 262-278.

[Gal] Galil, Z., "An Almost Linear Time Algorithm for Computing the Dependency Basis in a Relational Data Base," Res. Rept., Dept. of Mathematical Sciences, Computer Science Division, Tel Aviv University, Tel Aviv, Israel.

[GaJ] Garey, M. R., and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[HIT] Hagihara, K., M. Ito, K. Taniguchi, and T. Kasami, "Decision Problems for Multivalued Dependencies in Relational Databases," SIAM J. Computing, Vol. 8, No. 2 (May, 1979), pp. 247-264.

[HLY] Honeyman, P., R. E. Ladner, and M. Yannakakis, "Testing the Universal Instance Assumption," to appear.

[Kar] Karp, R. M., "Reducibility Among Combinatorial Problems," in Complexity of Computer Computations, (R. E. Miller and J. W. Thatcher, eds.), Plenum Press, New York, 1972, pp. 85-104.

[MMS] Maier D., A. O. Mendelzon, and Y. Sagiv, "Testing Equivalence of Data Dependencies," to appear in ACM Trans. on Database Systems.

[Ris] Rissanen, J., "Theory of Relations for Databases - A Tutorial Survey," Proc. 7th Symp. on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 64, Springer-Verlag, 1978, pp. 536-551.

[Sag] Sagiv. Y., "An Algorithm for Inferring Multivalued Dependencies With an Application to Propositional Logic," to appear in JACM.

[Yan] Yannakakis, M., private communication.

[Zan] Zaniolo, C., "Analysis and Design of Relational Schemata for

Database Systems," Tech. Rep. UCLA-ENG-7769, Dept. of Comp. Sci., UCLA, July, 1976.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-79-985	2.	3. Recipient's Accession No.
4. Title and Subtitle On the Complexity of Inferring Join Dependencies			5. Report Date August 1979
			6.
7. Author(s) David Maier, Yehoshua Sagiv*			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois			10. Project/Task/Work Unit No.
			11. Contract/Grant No. MCS-77-22830
12. Sponsoring Organization Name and Address National Science Foundation Washington, D.C.			13. Type of Report & Period Covered
			14.

15. Supplementary Notes

16. Abstracts

It is shown that deciding whether a set of functional dependencies and one join dependency implies another join dependency is NP-complete. It is also shown that deciding whether a JD-rule can be applied to a tableau T is NP-complete. This problem is NP-complete even if T can be obtained from a tableau corresponding to a join dependency by applying some FD-rules. As a result, it follows that computing the join of several relations is NP-hard.

17. Key Words and Document Analysis. 17a. Descriptors

functional dependency, multivalued dependency, join dependency, join, membership algorithm, NP-complete, relational database

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page UNCLASSIFIED	22. Price

FORM NTIS-35 (10-70)

USCOMM-DC 40329-P71